Embedding Nonlinear Optimization in RRT* for Optimal Kinodynamic Planning

Samantha Stoneman and Roberto Lampariello

***Abstract*— Some of the latest developments in motion planning methods have addressed the merging of optimal control with sampling-based approaches, to handle the problem of optimal kinodynamic motion planning for complex robot systems in cluttered environments. These include embedding the Linear Quadratic Regulator method in an RRT* context, or solving the kinematic problem with an RRT algorithm first and then feeding the solution to an NLP solver. An alternative approach is presented here, in which NLP is embedded in an RRT* context from the start. The resulting methodological features are illustrated with numerical examples. These include problems in which differential constraints play a fundamental role.**

I. INTRODUCTION

This paper addresses the robot trajectory planning task for systems which present nonlinear kinematic and differential constraints. To tackle the full problem, in which all constraints are treated simultaneously, two powerful but fundamentally different approaches are possible: the optimal control and the RRT* sampling-based methods. The first, when applied to realistic problems, typically leads to solving a nonlinear optimization problem via numerical iterative methods. These are becoming more and more popular in robotics [1] [2] [3] [4]. As is well known, they profit from gradient information and are efficient in making full use of the system's limits. However, they suffer, for the same reason, from risk of non-convergence, due to local minima. The latter are particularly abundant in robotic kinodynamic motion planning problems (see for e.g., [3]). In order to obviate this risk, gradient-based optimization solvers (referred to as GBOs here) are generally combined with global search methods, typically resulting in a Montecarlo search (see for example [3] [5]).

Given the good search space exploration capability of the RRT method [6], we argue that the latter is a better candidate for bringing the global search element into nonlinear-optimization-based motion planning methods, than a Montecarlo search is. It is primarily for this reason that we have developed a new motion planning algorithm in which a nonlinear optimization planner is embedded in an RRT* planner.

In this paper we present the algorithm for the resulting combined RRT*-GBO method, which we then analyze in some of its fundamental features. These include the efficient treatment of motion constraints, as well as particular newly introduced features, such as the possibility to converge to the goal quickly and exactly, pruning of the search tree and the treatment of system redundancy. Since we can link exactly to the goal, we gain substantially in the convergence rate to a solution, as well as introducing the possibility to efficiently handle redundant systems, for which the end goal configuration may not be given.

It is also interesting to view things from the perspective of the RRT*, which is known to benefit from a cost-to-go function when used as a metric for efficient tree propagation [7] [8]. Although the idea of implementing optimization in conjunction with RRT is not entirely new (see Section I-A), the optimization method is applied here to provide a solution for the tree edges which is geometrically and dynamically feasible at a preselected number of discretization points. The use of local gradient information results in an improved efficiency for solving the motion constraints, as opposed to RRTs. Furthermore, a smoothing algorithm, which is based on the same nonlinear optimization principle, is used to improve the otherwise jagged RRT* solution.

In our approach, the RRT* principally takes care of the problem topology (geometric and dynamic), providing the connectivity between homotopy groups, seen by the optimization method as local minima in which to look for a solution. The optimization then does that efficiently. In this way, the RRT* does not need to generate many nodes, as already shown in [9]. On this account, a parameter is introduced in the algorithm after which nodes that fall too close to existing nodes of the tree are discarded. This and other features are illustrated with numerical examples in simulation.

Due to the computational burden of this method, the long-term goal of our research scope is to follow the idea already presented in [3] [4], in which machine learning is used to provide optimal solutions, computed off-line, in a real-time setting. The piece of that idea which is presented here, aims at providing the efficient off-line generation of (near to) global optimal solutions for a given motion planning problem.

The paper is then structured as follows: the rest of Section I presents a literature survey and the problem statement, while Section II describes the formulation of the optimization problem and Section III the method for solving it. Section IV analyses the results in simulation and Section V presents the conclusions.

A. Related work

In [8] a similar kinodynamic planner formulation is presented, in which Linear Quadratic Regulation (LQR) is used to compute the edges of the RRT* tree, resulting in locally

The authors are with the Robotics and Mechatronics Center (DLR), 82234 Weßling, Germany, email: *firstname.lastname@dlr.de*

optimal kinodynamic trajectories. The method is limited to systems with linear dynamics, or with a local linearization of nonlinear dynamics. The same authors also recognize that LQR does not provide the means to handle constraints on the control inputs or on the states. The difference to our approach is therefore twofold: firstly, the linearization is only valid in a local region of the search space, which implies that many RRT* nodes are necessary to guarantee dynamic feasibility of the solution; secondly, constraints are handled by the RRT*, which does not make use of gradient information and is therefore computationally less efficient.

The idea of using an RRT solution as an initial guess for trajectory optimization was already proposed in [10], which is what we principally do in the smoothing task mentioned in section I. Important issues are pointed out in [10], with regards to the use of gradient-based nonlinear optimization methods, which are very often ignored in the literature, such as: the necessity of defining a suitable space of possible trajectories; the fact that trajectories not homotopic to the initial guess cannot be found and that in many cases variational techniques might not even find an optimal solution within a single homotopy class. In [7] the same author shows that an optimal metric for good propagation of the RRT tree is the cost-to-go. We address many of these points here and even show that there are motion planning problems for which the cost-to-go metric is not ideal.

[9] implements very similar concepts as done here, by combining RRT (not RRT*) with a Nonlinear Programming (NLP) solver for the computation of the edges. Many of the features described here are however absent in [9] extending to goal, pruning, rewiring with the RRT* approach, handling of motion constraints in the optimal control context), such that our work can be seen as an extension of it.

[11] and [12] solve the problem at hand in a two-step approach, in which the motion planning problem is first solved on a kinematic level with an RRT-like motion planner and the solution is then modified with a NLP solver to satisfy differential constraints. Note that this has not as general a scope as the one addressed here, since for some kinodynamic problems kinematic solutions may not be dynamically feasible and also not homotopic to a dynamically feasible solution.

A list of publications in which nonlinear optimization is applied to robot motion planning is given in the Introduction. In these, as well as in those mentioned above, relevant implemetation details can be found, also with respect to the treatment of collision avoidance constraints in the context of NLPs.

B. Problem statement

The kinodynamic motion planning problem of interest involves finding a trajectory from a given initial state to a given final state. This involves a search in a $2n$-dimensional state space, for n degrees of freedom. The final state may be a point in the state space or a region of it, if it refers to a final state in the operational space of a redundant system. Constraints are generally present on the states and on the input actions, in form of inequalities (to include collision avoidance to a given static obstacle region, given box constraints on position, velocity and actuator actions), and in form of equalities (to include given constraining dynamics). The trajectory should be feasible at a predefined number of points along it (the via points) and also optimize a predefined cost function generally of Lagrange type.

II. Formulation of the motion planning problem

The formulation of the motion planning problem of interest is a generalization of the one addressed in [3]. It typically contains a known obstacle region *O* and a bounded configuration space *C* of dimensions $C(\boldsymbol{\theta}) \subseteq \Re^n$ where $\boldsymbol{\theta}$ is the vector of the mechanical system degrees of freedom. The time interval is generally unbounded: $t = [0, \infty)$. The mechanical system is assumed here to be fully actuated and subject to a bounded action $\boldsymbol{\tau} \in \Re^n$, which is related to the system state $[\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}]$ by the state transition equation.

The nonlinear optimization problem can then be formulated as follows:

$$\min_{t_f, \boldsymbol{\theta}(t)} \Gamma(\boldsymbol{\theta}(t), \boldsymbol{\tau}(t), t_f) \tag{1}$$

subject to

$$\mathbf{h}(t_f, \boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t), \ddot{\boldsymbol{\theta}}(t)) = \boldsymbol{\tau}, \tag{2}$$

$$\mathbf{h}(t_f, \boldsymbol{\theta}(t)) \leq 0, \tag{3}$$

$$\mathbf{h}_{\text{coll}}(t_f, \boldsymbol{\theta}(t)) \leq 0, \tag{4}$$

$$\boldsymbol{\theta}(0) = \boldsymbol{\theta}_{\text{in}}, \dot{\boldsymbol{\theta}}(0) = 0, \dot{\boldsymbol{\theta}}(t_f) = 0. \tag{5}$$

$$\boldsymbol{\theta}(t_f) = \boldsymbol{\theta}_{\text{fin}} \text{ or } \mathbf{g}(\mathbf{r}^e(t_f), \boldsymbol{\phi}^e(t_f)) = 0, \tag{6}$$

for $0 \leq t \leq t_f$ and where t_f is the final time, Γ is a predefined cost function, $\mathbf{h}$ are inequality box constraints of type $\mathbf{x}_{\text{min}} \leq \mathbf{x}(t) \leq \mathbf{x}_{\text{max}}$, for $\mathbf{x} = \{\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\tau}\}$ and $\mathbf{h}_{\text{coll}}$ are collision avoidance constraints. Eq. (2) express the state transition equation of the system. Eq. (5) expresses boundary conditions on position, where $\boldsymbol{\theta}_{\text{in}}$ is the given initial configuration, and on velocity. The function $\mathbf{g}(\mathbf{r}^e(t_f), \boldsymbol{\phi}^e(t_f))$ is a set of equality constraints on the system state, which is only present for a redundant system, for example a redundant robot manipulator where $[\mathbf{r}^e, \boldsymbol{\phi}^e]$ represents its end-effector pose (see Section II-C). More details on the boundary conditions on acceleration and jerk will be given in Section III-C.

In the following we will address the formulations of the chosen cost function and of the inequality and equality constraints.

A. Cost function

The cost functions used here include the mechanical energy, the integral of the actuator actions and the Euclidean distance. The first two result in a cost-to-go function, which will be used as a metric in the RRT* algorithm to favour its search efficiency in state space [6]. The third will be used for means of comparison.

The mechanical energy is computed here as follows:

$$\Gamma_1 = \int_0^{t_f} (\boldsymbol{\tau}^T(t)\, \dot{\boldsymbol{\theta}}(t))^2\, dt\,, \tag{7}$$

expressing the integral of the power for the given system, while the integral of the actuator actions as:

$$\Gamma_2 = \int_0^{t_f} \boldsymbol{\tau}^T(t)\, \boldsymbol{\tau}(t)\, dt\,. \tag{8}$$

B. Inequality constraints

The bounds of the box constraints are given by the system design specifications. Further constraints arise from the collision avoidance both with the environment and of the system with itself (if applicable). To detect collision and to formulate the collision avoidance problem within a nonlinear programming context, bodies are represented here as convex polytopes. For these types of bodies, it is possible to efficiently compute, in case of collision, the penetration depth as the minimal length of translation needed to separate them.

The collision avoidance problem can be formulated straightforwardly as a set of inequality constraints in the optimization problem:

$$PD(i) \;\leq\; 0.0, \quad 1 \;\leq\; i \;\leq\; m_{\text{coll}}, \tag{9}$$

where the function $PD(i)$ constitutes a penetration depth between two intersecting bodies. The scalar m_{coll} is the number of body pairs in the given problem.

C. Equality constraints

Additional equality constraints may be required in the operational space of the system, e.g., on the final end-effector position and orientation [4] [3].

$$\mathbf{r}^e(t_f) - \mathbf{r}_{\text{des}}(t_f) = 0, \tag{10}$$

$$\boldsymbol{\phi}^e(t_f) - \boldsymbol{\phi}_{\text{des}}(t_f)) = 0, \tag{11}$$

where $[\mathbf{r}_{\text{des}} \quad \boldsymbol{\phi}_{\text{des}}](t_f)$ is the desired end-effector pose at the final time. These constraints are generally nonlinear in the system states. Note that this does not require specifying an end configuration of the system, if the latter is kinematically redundant with respect to the task.

III. Motion Planning Method of Solution

In this section the method to solve the optimization problem described above is addressed. This is based on the RRT* method, however with the formulation of its edges as boundary value problems solved as NLPs.

A. RRT* Algorithm

A salient description of the RRT* method can be found in [13]. We will limit ourselves here to describe the extensions which we brought to the method. Note that our implemented algorithm builds on [17], which was extended here with the following features:

1) GBO edges: As already mentioned, the edge generation is formulated as a BVP between the state boundaries defined by the two given tree nodes. The BVP is formulated as described in Section II above, with the first part of Eq. (6). This is solved as an NLP, as described in Section III-C.

2) Extend to goal: At the beginning of each iteration an edge to connect the last node in the tree to the goal is attempted, if the node is within a defined threshold of the goal is now used. Note that this kind of motion planning problem was typically solved in [3] [4] for fixed-based and free-floating robotic systems.

3) Vertices pruning: In order to limit the number of RRT* nodes and therefore the number of NLP function calls, a parameter is introduced which defines a threshold for the distance between new vertices and those existing in the tree. If this distance is below the threshold, the sampled node is abandoned and a new iteration begins. A constant pruning threshold is undesirable because the free space can become *saturated* with nodes as shown in Section IV, Fig. 4. Thus a dynamic r_{prune} is computed for each iteration based on a sliding window of the current tree state. An initial radius $r_{initial}$ provided by the user is scaled as a function of a window size N_{window} and difference of successful nodes in the window and pruned nodes in the window, $N_{pruned,window}$.

$$r_{prune} = r_{initial} \frac{1}{\exp(N_{window} - (N_{pruned,window})} \tag{12}$$

4) RRT-GBO Algorithm:* The resulting algorithm is described in the pseudo-code below. There are four main segments, (1) sampling, pruning and initial edge creation, (2) neighborhood search from the RRT* algorithm, (3) rewire phase from the RRT* algorithm, and (4) extension to goal. The attempt of reaching the goal is based on the configurable proximity variable $r_{connect}$. The tree can be made as sparse or dense as the user would like by the configurable pruning variable r_{prune}.

Algorithm 1 *RRT^*-GBO Iteration*

for $iter \rightarrow N$ **do**
  if $r_{connect} \geq \|x_{goal} - x_{last}\|$ **then**
    $connect\_to\_goal(x_{last}, x_{goal})$
  $x_{new} \leftarrow rand()$
  if $x_{new} \notin \mathcal{R}_{obstacle}$ **then**
    $x_{near} \leftarrow kdtree(x_{new})$
    if $r_{prune} \leq \|x_{near} - x_{new}\|$ **then**
      $e_{near \rightarrow new} \leftarrow gbo\_extend(x_{new}, x_{near})$
      $add\_to\_tree(e_{near \rightarrow new}, x_{new})$
      for $size(\mathcal{T})$ **do**
        if $\|x_i - x_{new}\| \leq r_{ball}$ **then**
          $neighborhood\_query(x_{new}, x_i)$
        if $\|x_i - x_{new}\| \leq r_{ball}$ **then**
          $rewire\_query(x_i, x_{new})$

Algorithm 2 $connect\_to\_goal(x_{last}, x_{goal})$

if $e_{last \to goal} \leftarrow gbo\_extend(x_{goal}, x_{last})$ **then**
$add\_to\_tree(e_{last \to goal}, x_{goal})$

Algorithm 3 $neighborhood\_query(x_i, x_{new})$

$e_{i \to new} \leftarrow gbo\_extend(x_{new}, x_i)$
if $cost\_to\_go(e_{i \to new}) < cost\_to\_go(e_{near \to new})$ **then**
$remove\_from\_tree(e_{near \to new})$
$add\_to\_tree(e_{i \to new})$

Algorithm 4 $rewire\_query(x_i, x_{new})$

$e_{new \to i} \leftarrow gbo\_extend(x_i, x_{new})$
if $cost\_to\_go(e_{new \to i}) < cost\_to\_go(e_{i_{incoming}})$ **then**
$remove\_from\_tree(e_{i_{incoming}})$
$add\_to\_tree(e_{new \to i})$

B. RRT Metric*

As is well known, the metric for the RRT algorithm is important for an efficient tree growth and search space exploration. The metric is in fact used to compute all distances of relevance (nearest node, rewiring area, pruning area, connect to goal). We chose here the cost functions defined in Section II-A. The NLP method however allows formulating any cost function of interest, as long as this is C^2.

We also want to address an interesting issue, which relates to how the "nearest node" to a new node is selected. In our formulation, the computation of the distance of a new node to those already existing in the tree, results from the solutions of the relative generally nonlinear BVPs. Because of the nonlinear nature of these problems, there is a risk that some of these computations fall into local minima and not into the global minimum (depending on the initial guess used, multiple solutions for the distance can be found). As such, the resulting distance computation will generally be biased by this detrimental effect. This is however removed by the rewiring process of the RRT* and by a prolonged search for a global solution. We therefore generally do not look for the first feasible solution, but rather for a (near to) global one, or practically the one found in an acceptable computational time.

Note that the rewiring feature is in this way fundamental, since it makes the RRT* solutions improve towards the global optimum. This problem characteristic however stems from the nonlinear nature of the kinematics and dynamics of the mechanical systems of interest, which can only be dealt with through the use of a method which has probabilistic completeness properties. The pruning parameter also has a large influence on this effect, since the further apart the nodes, the greater the role of the nonlinearities. More will be said about this point in Section III-C and in Section IV.

C. Steering method - Local optimizer

The steering problem consists of solving the BVP formulated through Eqs. (1)- (5). The chosen method of solution is such that the system states are parameterized in time with B-splines of order four, to guarantee smoothness up to the second derivative, and with N_{vertex} parameters per state. The boundary conditions are then solved implicitly by a direct algebraic computation of the B-spline parameters. The minimum value of N_{vertex} is then nine, to allow for a minimum of one degree of redundancy for the shape of the trajectory between the fixed boundaries. The optimal number of Bspline parameters however is strongly function of the problem at hand. Empirical results in [3] have shown that a 50% improvement can be achieved in the optimization result with a suitable parameterization and number of parameters. This might also be affected by the presence of saturated solutions (e.g., in velocity), which B-splines can represent very well.

The optimization problem described above is solved as an NPL, by satisfying the equality and inequality constraints at a finite number of k via points. The method of solution is the one described in [3] for fully actuated systems. The state transition equations are trivially solved for any parameterization of the states and the respective control actions are computed via the inverse dynamics expressed in Eq. (2).

The resulting NPL is solved with the Sequential Quadratic Programming algorithm from the Nlopt library by [16]. To compute the penetration depth between two bodies, the Bullet Physics library by [15] was used. The library allows representing objects as boxes or capsules. Each pair of intersecting objects is treated separately and penetration depth can be evaluated for each pair straightforwardly.

The duration in time and density of via points of the edges in the results presented here are fixed. The edge optimization is performed with just one free Bspline coefficient. Additionally, the acceleration and jerk bound conditions for each edge are fixed to zero. These fixed settings yielded good results, presented in section IV; however, the settings can be easily tailored within the algorithm implementation if a scenario requires it.

The NLP solver is given a maximum number of iterations to ensure that the total running time remains upper bounded.

D. Smoothing method

The RRT*-GBO solution, consisting of N_{RRT} nodes and intermediate via points, is approximated as a B-spline where the boundary (root and goal) positions, velocities, accelerations and jerks are constrained with four fixed Bspline coefficients each and the intermediate positions are the free Bspline coefficients. The RRT*-GBO solution trajectory can be sampled such that $N_b = N_{RRT}$ The resulting Bspline is set as the initial guess and is smoothed using the same GBO optimization routine as computes the RRT edges. This results in an initial guess which is homotopic to the optimal solution to be found. Multiple RRT* solutions can be computed, as these will progressively improve (locally and globally), and are fed to the smoothing algorithm.

IV. Analysis of Results

We present in the following section a series of benchmarks and scenarios which demonstrate the capabilities of the new embedded method and the implemented algorithm RRT*-GBO. The scenarios include a 2DOF translating problem in a cluttered space, previously computed as a benchmark in [8], a 3DOF scenario using orbital dynamics in a cluttered environment as previously computed in [14] and finally a redundant double pendulum swing up scenario. Additionally, two tests are computed to show the behavior of the important features which define the RRT*-GBO algorithm, (1) dynamic pruning and (2) connection to goal, as well as a solution comparison to a basic RRT* implementation.

A. Example 1: Goretkin's benchmark problem

The embedded method was benchmarked on a 2DOF simple translating scenario that was previously solved in [8]. The Goretkin benchmark problem is a point-to-point problem for a simple point mass. The problem formulation was implemented for RRT*-GBO with the following formulation: minimize the cost function of Eq. (8), such that,

$$\begin{aligned}
\mathbf{M}\ddot{\mathbf{r}} &= \mathbf{f}, \\
\mathbf{x}_0 &= \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T, \\
\mathbf{x}_f &= \begin{bmatrix} 8 & 0 & 0 & 0 \end{bmatrix}^T, \\
-2 \leq r_x(t) &\leq 10, \\
-8 \leq r_y(t) &\leq 8, \\
\|\dot{\mathbf{r}}(t)\| &\leq \begin{bmatrix} 2.5 & 2.5 \end{bmatrix}^T, \\
\|\mathbf{PD}(t)\| &\leq 0,
\end{aligned}$$

where,

$$\begin{aligned}
\mathbf{x} &= \begin{bmatrix} \mathbf{r} & \dot{\mathbf{r}} \end{bmatrix} \\
\mathbf{M} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned}$$

Here $\mathbf{M}$ is the mass matrix and $\mathbf{x}$ is the four dimensional state vector. In this case boundary constraints were not imposed on the external input $\mathbf{f}$ to maintain coherence with the Goretkin implementation.

The RRT*-GBO algorithm was run only until one solution was found. The RRT* method within RRT*-GBO is extremely useful in that as the RRT* progresses through many solutions, the *cost to go* is improved through the backwards optimizing characteristic of the rewire function. However, this problem is linear, with low dimensionality and is therefor extremely easy to solve for the modified RRT*-GBO algorithm. Thus the first solution although not optimized is good enough because the final smoothing phase can easily converge to the *near* global minimum. The GBO accuracies were relaxed for the edge creation and initial smoothing, such that $\eta_g = 1e^{-5}$ and $\eta_J = 1e^{-4}$ and were then tightened for the final smoothing such that $\eta_J = \eta_g = 1e^{-6}$, where η_g is the gradient finite differencing computation step size and η_J sets the stopping value for the absolute minimum change in cost function.

Table I presents the results of the first RRT*-GBO solution trajectory and its two successive smoothings as well as a solution using an implementation of a basic RRT* and edges parameterized with cubic Bsplines. The RRT* benchmark solution was iterated until 15 successively improved solutions were reported, which required over 1000 iterations.

It can be seen immediately that the connect to goal feature and GBO of the edges drastically improve the solution. In this case the RRT*-GBO solution computation time is more that 500% improved over the best RRT* solution. Although the RRT* is able to find an initial solution in only 414 iterations, the resulting cost to go is not optimized in comparison.

The RRT*-GBO and RRT* solutions are shown in figures 1 and 3. Fig. 2 shows the evolution of the pruning

	N_{iter}	N_{RRT}	Cost [N^2]	Time [s]
RRT*-GBO	103	16	909.3	0.239
1st Smoothing	-	-	46.3	1.505
2nd Smoothing	-	-	9.0	8.651
Basic RRT* 1st	414	910	1100	0.599
Basic RRT* 15th	1349	910	104.8	5.466

TABLE I: Goretkin benchmark solutions summary. The initial RRT*-GBO solution found + 2 smoothings are compared to the initial solution and 15th solution of the same scenario using a basic RRT* and Bspline trajectory parameterization without the pruning, connect to goal or GBO edge features.

radius applied to the Goretkin benchmark problem solution. The pruning radius is a function of the number of iterations, or node attempts, and the number of those attempts which fail due to pruning as described in Eq. (12). In this way the pruning radius dynamically increases or decreases based on the saturation of the space being explored. In this case,

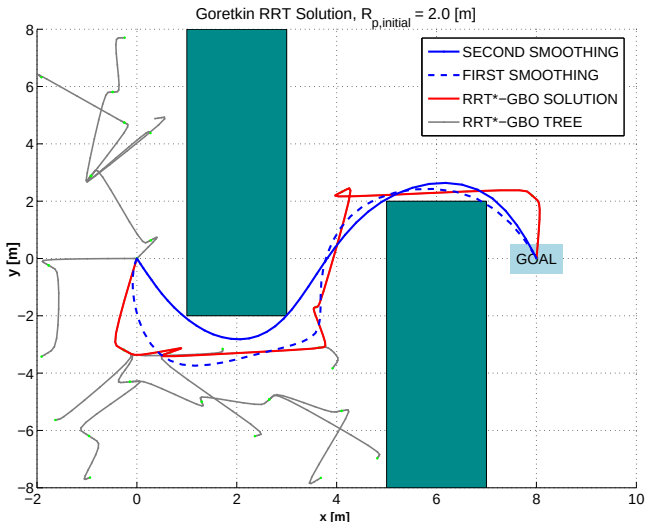

Fig. 1: RRT*-GBO + smoothing solution to Goretkin benchmark problem. The solution before smoothing is shown in red, and the successive smoothing results are shown as the dotted and solid curves respectively.

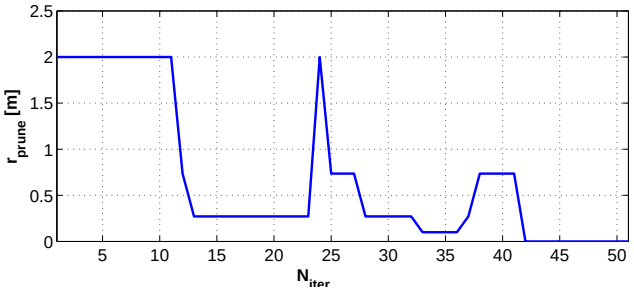

Fig. 2: Evolution of pruning radius r_{prune} vs. iterations for the RRT*-GBO Goretkin Benchmark solution.

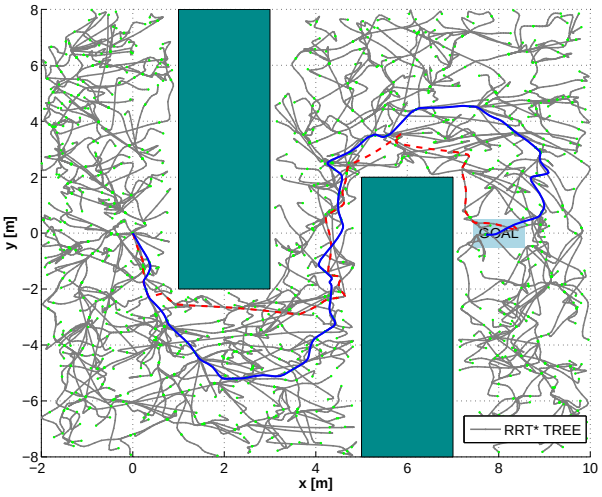

Fig. 3: Solution to Goretkin benchmark using a basic RRT*. The RRT* was stopped after 15 solutions had been found, the initial solution is shown in in red and the final (15th) solution is shown as the solid blue curve.

when the corridor region is saturated with nodes, the radius decreases and new nodes are allowed to be built. As the planner moves into the then open space on the goal side of the corridor the radius can be increased again to allow for more efficient searching.

B. RRT-GBO salient feature tests*

The Goretkin problem was attempted with the pruning radius dynamic increasing and decreasing disabled, i.e. constant pruning radius. The resulting tree is shown in Fig. 4. The problem was again attempted with the connect to goal feature disabled, the resulting tree is shown in Fig. 5.

C. Example 2: 3 dof body in orbit around ISS and/or Envisat

A comparison was made to a very different method, namely the mixed integer linear programming method in [14], on a point to point motion planning problem for a micro satellite maneuvering around modules of the ISS.

The Hill dynamics are used for the satellite and the ISS and thus the problem has 3DOF and a 6 dimensional state space. The cost function was taken to be the sum of the required applied forces, as in Richards formulation, and is

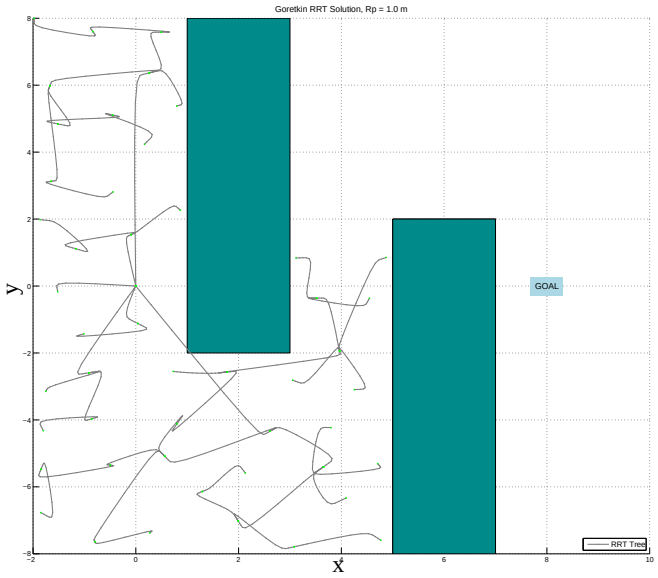

Fig. 4: RRT*-GBO tree in a 2DOF scenario with constant pruning radius $r_{prune} = 1.0$. This tree resulted after 1000 iterations, no solutions were found due to saturation in the corridor region.

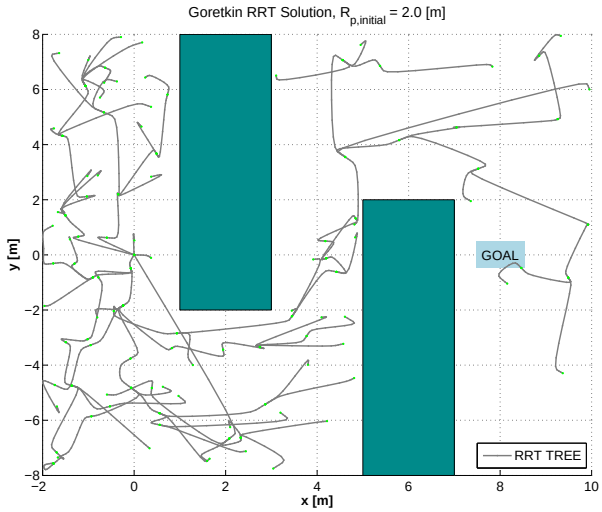

Fig. 5: RRT*-GBO tree in a 2DOF scenario without the connect to goal feature. This tree resulted after 1000 iterations, no solutions were found due to the fact that no nodes fell in the goal region (shown in light blue).

described in Eq. (8). The formulation of the constraints were as follows. Minimize Eq. (8) subject to,

$$\ddot{x} + 2n\dot{y} - 3n^2 x = f_x$$

$$\ddot{y} + 2n\dot{x} = f_y$$

$$\ddot{z} + n^2 z = f_z$$

and,

$$\mathbf{q}_0 = \begin{bmatrix} 10 & 10 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{q}_f = \begin{bmatrix} 5 & -5 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$\|\mathbf{f}_{ext}(t)\| \leq 1.0mN,$$
$$\|\mathbf{q}(t)\| \leq \begin{bmatrix}20 & 20 & 20 & 0.25 & 0.25 & 0.25\end{bmatrix}^T$$
$$\|\mathbf{PD}(t)\| \leq 0,$$

given,

$$m_{sat} = 5\ kg$$
$$T_{orbit} = 90\ min.$$

A summary of the results is shown in Table II. In order to provide a comparison to the solution in [**?**], the cost here is shown in units of velocity change, Δv. The RRT*-GBO results and solution shown in Fig. 6 represent the 5$^{\text{th}}$ solution found after building 1561 nodes in the search space. The RRT*-GBO achieved a better cost than the best MILP solution with a computation time less than 50% of the best result from [14] .

Of interest to note is that due to the orbital dynamics and inherent harmonics, the minimal cost to go solutions are not the shortest path solutions, thus an orbital dynamics scenario such as this presents a situation where the euclidean metric is not ideal. Additionally, the problem is also difficult to compute when the RRT metric is chosen to be the cost to go. In this case the pruning feature will hinder the solution building by pruning away force-free solutions.

D. Example 3: double inverted pendulum

Finally the results of the RRT*-GBO algorithm are shown applied to a redundant, fully actuated double pendulum system. This scenario involved 2DOF, one for each joint position, and thus the state space was 4 dimensional to include the positions and velocities of the joints. For this solution the cost function and RRT* r_{ball} metric were computed as the joint energy squared. The formulation of the problem was a follows:

$$\min_{\mathbf{p}} \quad \Gamma_{\text{energy}}(\mathbf{p}) = \sum_{0}^{t_f} (\boldsymbol{\tau}^T(t,\mathbf{p})\,\dot{\boldsymbol{\theta}}(t,\mathbf{p}))^2\,, \qquad (13)$$

such that,

$$\mathbf{M}\ddot{\theta} + \mathbf{C}\dot{\theta}^{\mathbf{2}} + \mathbf{g} = \tau,$$
$$\mathbf{q}_0 = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T,$$
$$\|\mathbf{q}(t)\| \leq \begin{bmatrix}\frac{6\pi}{5} & \frac{3\pi}{4} & \frac{3\pi}{2} & \frac{3\pi}{2}\end{bmatrix}^T,$$
$$r_{ee,x} = 0,$$
$$r_{ee,y} \geq 0$$

	Cost [Δv]	**Time [s]**
RRT*-GBO	1.2851	18
First Smoothing	0.2447	308
Second Smoothing	0.1906	412
Richards et. al. [14]	0.2692	1800

TABLE II: ISS Scenario summary. Computation time is shown in seconds and the cost was converted to total change of velocity.

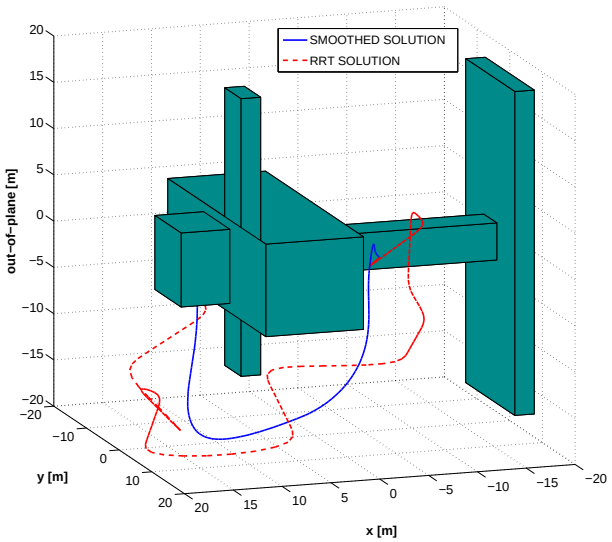

Fig. 6: A RRT*-GBO + smoothing solution to the ISS maneuver scenario.

given:

$$\mathbf{M} = \begin{bmatrix}(m_1+m_2)l_1^2 & m_2 l_1 l_2\cos(\theta_2-\theta_1)\\ m_2 l_1 l_2\cos(\theta_2-\theta_1) & m_2 l_2^2\end{bmatrix}$$
$$\mathbf{C} = \begin{bmatrix}0 & (m_2 l_1 l_2\sin(\theta_2-\theta_1)\\ -m_2 l_1 l_2\sin(\theta_2-\theta_1) & 0\end{bmatrix}$$
$$\mathbf{g} = \begin{bmatrix}l_1(m_1+m_2)g\\ l_2 m_2 g\end{bmatrix}\sin(\theta),$$
$$l_1 = l_2 = 2\text{m},$$
$$m_1 = m_2 = 4\text{kg},$$
$$g = -9.81\text{m/s}.$$

A brief summary of the results first from the RRT*-GBO algorithm and after a single smoothing is presented in Table III. The GBO edges and smoothing were evaluated with equal function accuracy and gradient step size, such that $\eta_J = \eta_g = 1e^{-6}$. In this case loose constraints were set on the actuation, well above the necessary torque to achieve the static horizontal position, to simulate an unconstrained scenario.

It is clear from the result that an attempt was made to connect the root node to goal, which was not defined but rather constrained by the equality and inequality constraints on the end effector position. This attempt however failed due to the constraints on position, which are a necessary characteristic of the RRT. The second iteration was than successful and brought the arm to lowest possible position and thus the state with the lowest potential energy. The final end effector r_y position was constrained by the bound constraints on the joint states and from the figure the second joint goes exactly to the limit. The pendulum positions over time are shown for the RRT*-GBO and smoothed solution in Figure 7 below.

	N_{iter}	N_{RRT}	**Cost [(N/s)^2]**	**Time [s]**
RRT*-GBO	2	2	12340	0.063
after smoothing	-	-	186.5	2.151

TABLE III: Pendulum benchmark summary

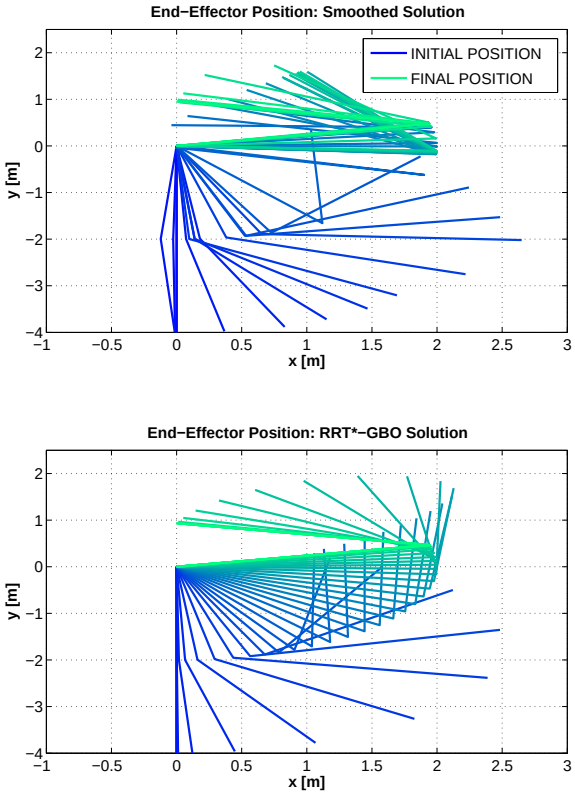

Fig. 7: Double pendulum solution with redundancy in the final position. RRT*-GBO initial solution is shown in bottom panel and and the resulting smoothed solution is shown in the top panel.

V. Conclusion

In conclusion, we presented a new algorithm for solving general kinodynamic motion planning problems, based on an extended RRT* algorithm, in which the edges are formulated as nonlinear optimization problems. This allows to address systems with nonlinear kinematics and dynamics, as well as optimizing general smooth cost functions and adding algorithmic features which enhance the planner's computational efficiency.

Future work will see the implementation of this algorithm for realistic scenarios as well as for underactuated, nonholonomic systems.

VI. Acknowledgements

This work was funded by the Bavarian Research Foundation in the context of the FORROST project.

References

[1] Koch, K., Mombaur, K., Soueres, P., "Optimization-based walking generation for humanoid robot", 10th IFAC Symposium on Robot Control, 2012.

[2] S. Miossec, K. Yokoi and A. Kheddar, "Development of a software for motion optimization of robots - Application to the kick motion of the HRP-2 robot", *Proceedings of IEEE International Conference on Robotics and Biomimetics*, 2006.

[3] Lampariello, R., Nguyen-Tuong, D., Castellini, C., Hirzinger, G., Peters, J., "Trajectory planning for optimal robot catching in real-time", ICRA 2011, China, May 2011.

[4] Lampariello, R., Hirzinger, G., "Generating Feasible Trajectories for Autonomous On-Orbit Grasping of Spinning Debris in a Useful Time", IROS 2014, Tokyo, Japan, November 2014.

[5] Park, J., Bobrow, J.E.: *Minimum-Time Motions of Manipulators with Obstacles by Successive Searches for Minimum-Overload Trajectories*, ICRA 02, Washington, USA, May 2002.

[6] La Valle, S., Kuffner, Jr., J.J., "Randomized Kinodynamic Planning", *The International Journal of Robotics Research*, Vol. 20, No. 5, 378-400 (2001).

[7] Cheng, P., La Valle, S., "Reducing Metric Sensitivity in Randomized Trajectory Design", IROS 2001, Hawaii, USA, 2001.

[8] Goretkin, G., Perez, A., Platt, R., and Konidaris, G. *Optimal sampling-based planning for linear-quadratic kinodynamic systems*. In Robotics and Automation (ICRA), 2013 IEEE International Conference on (May 2013), pp. 2429-2436.

[9] Karatas, T., Bullo, F., "Randomized searches and nonlinear programming in trajectory planning", CDC 2001.

[10] La Valle, S., "Planning Algorithms", Cambridge University Press, pp. 852, 2006.

[11] Bouffard, P., Waslander, S., "A Hybrid Randomized/Nonlinear Programming Technique For Small Aerial Vehicle Trajectory Planning in 3D", IROS 2009.

[12] El Khoury, A., Lamiraux, F., Taiex, "Optimal Motion Planning for Humanoid Robots", ICRA 2013.

[13] Karaman, S., Frazzoli, E., "Sampling-based Algorithms for Optimal Motion Planning", *The International Journal of Robotics Research*, vol. 30, iss. 7, pp. 846-894, 2011.

[14] Richards, A., S. T. H. J. P., and Feron, E. "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming". *Journal of Guidance, Control, and Dynamics* 25, 4 (2002).

[15] Coumans, E. Bullet 3d real-time multiphysics library. *http://bulletphysics.org/wordpress*.

[16] Johnson, S. G. The nlopt non-linear optimization package. *http://ab-initio.mit.edu/nlopt*.

[17] Karaman, S., and Frazzoli, E. *http://sertac.scripts.mit.edu/rrtstar*.